Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2343 - Arquitectura de Computadores

Tarea 1

15 de marzo de 2019

Entrega: 21:59:59 del viernes 29.03.2019

1º semestre 2019 - **Profesor:** Yadran Eterovic

Requisitos

- Esta tarea es estrictamente individual. Cualquier tipo de falta a la honestidad académica será sancionada con la **reprobación** del curso con la nota mínima.
- Los ejercicios de tipo **programación** deberán ser realizados en `Python` 3.6.X.
- Los nombre de archivos y el cómo deben ser ejecutados son parte del formato, no respetarlo será penalizado.
- Los ejercicios de tipo **placa** deberán ser realizados en `VHDL`.
- Los ejercicios de tipo **teóricos** deberán ser contestados en un archivo `Markdown` y subirlo junto a su tarea, de nombre Respuestas.md, en el mismo repositorio.
- Toda tarea debe estar acompañada de un README.md que explique de forma clara el funcionamiento de la misma.
- Esta tarea deberá ser subida a su repositorio personal de *GitHub* correspondiente en la fecha y hora dada.

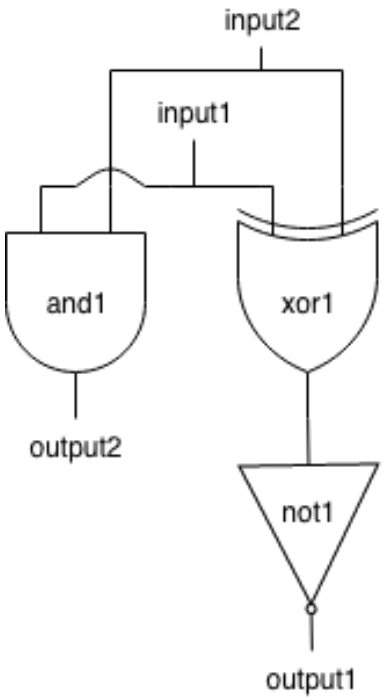
Representación de circuitos lógicos

La descripción de un circuito estará contenida en un archivo `JSON` respetando el siguiente formato: `[{"from": xxxx, "to": xxxx}, ..., {"from": xxxx, "to": xxxx}]`. Es decir, una lista de diccionarios, donde cada diccionario representa una conexión entre dos elementos del circuito. Dichos elementos pueden ser entradas, salidas o compuertas lógicas.

La convención que se empleará para las entradas y salidas será de la forma `input{i}` y `output{j}`, respectivamente. Análogamente, la convención para las compuertas será de la misma forma, es decir, `and{k}`, `or{l}`, `xor{m}`, `not{n}`, `nand{o}` y `nor{p}`. **No deben asumir un orden específico para las compuertas**, pero sí que los archivos contendrán circuitos válidos.

A continuación se da un ejemplo simple para mostrar la manera correcta de emplear dicha descripción. Esta corresponde al circuito expuesto en el diagrama.

```
[
    {
        "from": "input1", "to": "xor1"
    },
    {
        "from": "input1", "to": "and1"
    },
    {
        "from": "input2", "to": "xor1"
    },
    {
        "from": "input2", "to": "and1"
    },
    {
        "from": "xor1", "to": "not1"
    },
    {
        "from": "not1", "to": "output1"
    },
    {
        "from": "and1", "to": "output2"
    }
]
```



Ejercicios - Programación

1. Equivalencia de circuitos

Escribe un programa `logic_translator.py` que, dada la descripción de un circuito lógico binario compuesto únicamente por las compuertas `AND`, `OR`, `XOR` y `NOT`, genere la descripción de un **circuito equivalente**[1] compuesto únicamente por compuertas `NAND` o únicamente por compuertas `NOR`.

El archivo a ejecutar debe recibir tres argumentos **por línea de comando**: el *path* del archivo[2] que describe el circuito original, el *path* del archivo en el que se escribirá el circuito equivalente y el string `nand` o el string `nor` (en ese orden) — indicando, de esta manera, si se desea obtener el circuito formado por compuertas `NAND` o por compuertas `NOR`, respectivamente.

La imagen de a continuación ejemplifica la manera correcta de ingresar los argumentos a través de la consola utilizando un archivo `main.py`

```
C:\Users\User>py -3.6 logic_translator.py original_circuit.json equivalent_circuit.json nor
```

2. Simulación de circuitos

Crea un programa `circuit_simulator.py` que, dada la descripción de un circuito lógico binario compuesto únicamente por las compuertas `AND`, `OR`, `XOR` y `NOT`, y dados los *inputs* de dicho circuito, indique los valores de salida de todas las compuertas lógicas que lo componen y de todos los *outputs* que en él se encuentran.

Al igual que en el problema anterior, el archivo a ejecutar debe recibir **dos** argumentos **por línea de comando**: el *path* del archivo que describe el circuito con sus inputs y el *path* del archivo que en el que se escribirá el resultado (en ese orden).

[1] Dos circuitos son equivalentes si dado el mismo *input*, generan el mismo *output*.
[2] Recuerda que la extensión de un archivo es parte de su nombre.

Ejemplo simulación

Supongamos que este ejemplo tiene el archivo `JSON` que describe el siguiente circuito. Es decir, un diccionario, en el que cada elemento representa uno de los *inputs* del circuito, al que le corresponde un `0` o un `1` como valor, junto a su circuito asociado. Asimismo, los valores de salida deberán ser escritos siguiendo el mismo formato.

```
{
  "input": {
            "input1":0,
            "input2":1
  },
  "circuit":[
  {"from": "input1", "to": "not1"},
  {"from": "input2", "to": "not2"},
  {"from": "not1", "to": "and1"},
  {"from": "not2", "to": "and1"},
  {"from": "and1", "to": "output1"}
  ]
}
```

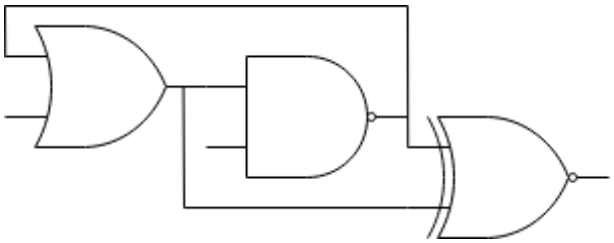
Se espera que como *output* tu programa genere un archivo `JSON` como el mostrado a continuación:

```
{
  "not1": 1,
  "not2": 0,
  "and1": 0,
  "output1": 0
}
```

Ejercicios - Teórico

1. ¿Circuitos con ciclos?

Muchos circuitos lógicos que se utilizan en la realidad contienen ciclos, es decir, existen conexiones entre la salida de una compuerta lógica y una de sus entradas, ya sea directa o indirectamente. La imagen que se encuentra a continuación ejemplifica esta situación.



Dada esta situación de ciclos, ¿qué tipo de comportamiento tendría un circuito con ciclos? ¿Qué usos se le podría dar? Comente con un límite de 500 palabras en total y mencione los factores a tener en cuenta al hacer uso de estos circuitos. Se recomienda adjuntar imágenes, dibujos y/o diagramas para apoyar su explicación. **HINT: Piense en los *outputs*.**

Entrega y evaluación

La tarea se debe realizar de **manera individual** y la entrega se realizará a través de GitHub. Archivos que no compilen y/o que no cumplan con el formato de entrega implicarán nota **1.0** en la tarea. En caso de atraso, se aplicará un descuento de **1.0** punto por cada 6 horas o fracción.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.